Team Controller
Northern Arizona University
Flagstaff, Arizona
October 20th, 2023

Zachary Parham (Team Lead): zjp29@nau.edu
Italo Santos (Mentor): ids37@nau.edu
Bradley Essegian: bbe24@nau.edu
Brandon Udall: bcu8@nau.edu
Dylan Motz: djm658@nau.edu

Requirements Specification

Northrop Grumman

Weapon System Support Software

Harlan Mitchell

Laurel Enstrom

Accepted as baseline requirements for this project by:

Client :	____________	____________
Team Lead :	____________	____________
	Date	Signature

Table Of Contents

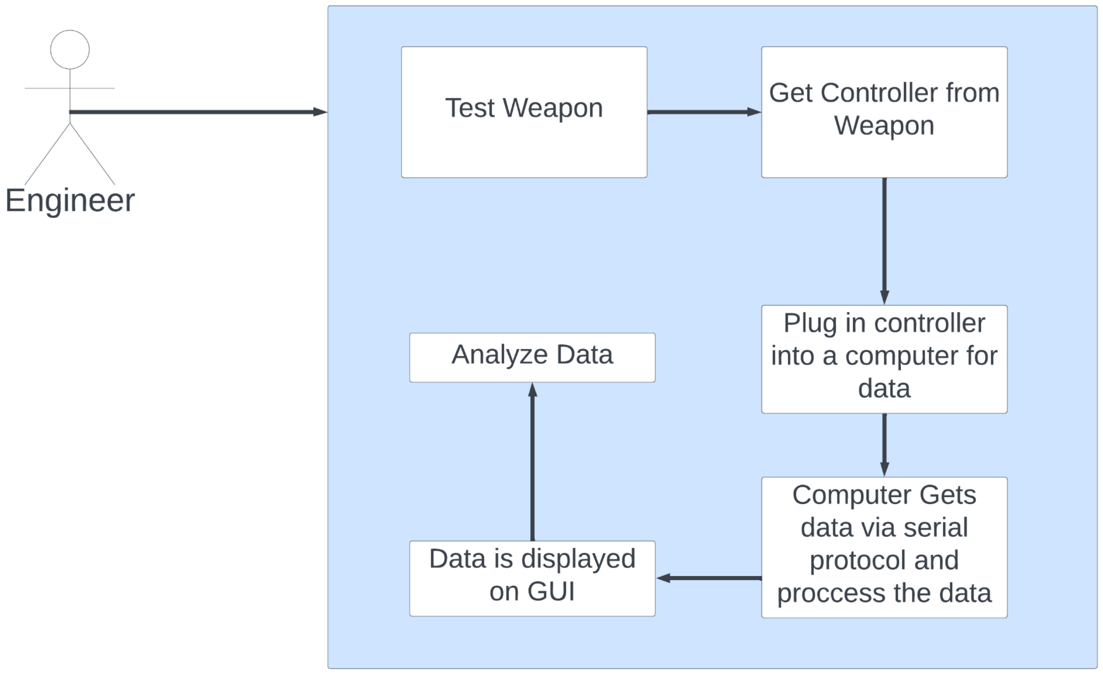
1.0 Introduction

In the realm of aerospace and defense technology; where precision, security, and reliability are of utmost importance, the need to effectively diagnose and rectify issues with military weaponry is paramount. Multi-billion dollar defense contractors are equipping the United States with products ranging from advanced military aircraft to state-of-the-art weaponry. This defense sector plays a vital role in national security, global stability, and is substantial in terms of economic impact. Every year these systems become increasingly more complex and must stay reliable.

Northrop Grumman, our clients, stand as a prominent figure in the aerospace and defense technology industry. A cornerstone in the American military, they have built everything from the B-2 Stealth Bomber to the James Webb Space Telescope [1, 2]. It's no surprise that these revolutionary projects create an endless amount of diagnostic data. With their formidable presence, they provide a spectrum of armament systems and services to their customers in over 25 nations around the world [3]. Northrop Grumman's success is not solely measured in monetary terms; they play a critical role in the national security and well-being of military personnel worldwide. The heart of the challenge that our clients face lies in the diagnostics of armament systems. Sr. Systems Engineer Manager, Harlan Mitchell, and Principal Systems Engineer, Laurel Enstrom, are faced with a cumbersome and resource-intensive process for obtaining diagnostics on their products when the end-user has a problem. This document will outline the requirements for our software that shall transform this process, enabling maintenance personnel and their customers to easily obtain the best diagnostic data possible regarding the weapon system. In today's climate, ensuring that our military armament systems are functioning optimally is not just a matter of national security; it is a global concern.

2.0 Problem Statement

Our clients need to test weapons that are developed by many different engineers and to do this they utilize a controller that gets the weapons data and a software product to process this data. The clients have many issues with this software product since it's complicated. The process of how the software product they currently have works like this.


Engineer
Test Weapon
Get Controller from Weapon
Plug in controller into a computer for data
Computer Gets data via serial protocol and proccess the data
Data is displayed on GUI
Analyze Data

At Northrop the current issue with the software product they currently have is that it is complicated so that only engineers can understand the program and the people at the test sites don't know how to use the software. Also sending a team of engineers down to the test site to test a weapon every time takes away time and resources from other engineering tasks. Finally it also needs admin permissions to install the software.

The current inefficiencies and missing elements in the client's current software product:

- Very complicated GUI that only engineers can understand
- No way for test site workers to use/communicate the data back to the engineers in Phoenix
- Needs admin permission to install

3.0 Our Solution: A Graphical Data Display

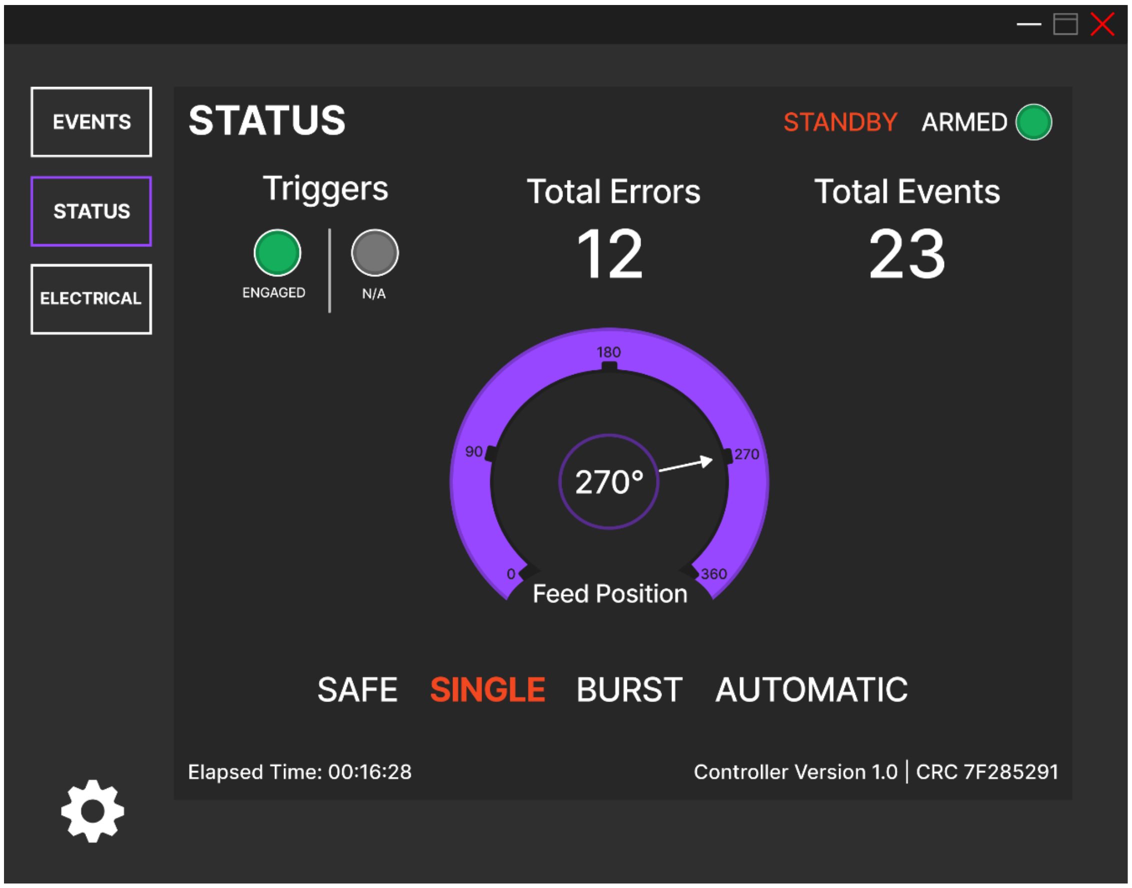
To help our clients diagnose problems within their weapon systems, we propose a desktop application which can read serial data sent from a weapon controller and display it in a user friendly way. The weapon controller is already capable of generating and sending data such as firing mode, firing rate, weapon position (in degrees) etc in the form of RS422 serial messages. Our application must be able to read this data and use it to generate a comprehensive graphical report which will keep weapon operators, engineers and technicians informed on the current status of their system.

Our application will display the following data:

- Weapon status (armed or disarmed)
- Serial baud rate
- Record of errors encountered since weapon activation (overheat, jam, mechanical failure, etc.)
- Record of events logged since weapon activation (firing events, reload events, etc.)
- Time elapsed since weapon activation
- Current Firing mode (automatic, burst, single, safe)
- Current burst length (number of concurrent launches per firing event)
- Current firing rate (number of launches per second weapon is capable of)
- Estimated feed position (weapons current state in firing cycle i.e. ejecting spent shell, feeding new shell, firing etc.)
- Electrical components (a list of various components including voltages and currents for each)

Some categories of data such as feed position and trigger status can be represented graphically. Other data categories such as records of errors and events will need to be logged in text format. Our goal is to find the most intuitive and simple way of representing each data category. We must also streamline user navigation between different menus and data displays. This will make our application easy to learn and allow users to quickly gain access to any piece of information we track.

Below is a sample rough draft design of the status section of our GUI. This graphic is subject to change as development of this project progresses.



Additionally, our project must implement cybersecurity concepts to eliminate the possibility of misuse. This means reducing the level of user interactions with our system to very predictable and well guarded inputs most often in the form of buttons and dropdown menus.

Our envisioned software product will lead to more effective weapon operation and understanding whether that be in the field or during testing/repair.

4.0 Project Requirements

In this section we will introduce specific requirements that our finished product must satisfy. These requirements must be verifiable or in other words we must be able to measure some aspect of our product to undeniably prove that they are satisfied.

The names of each requirement ex. CR01 (client requirement 01) and SR01 (software requirement 01) represent their scope. To put it simply, client requirements are the requirements we derived directly from the needs of our clients. They represent things the client cares about. Updates given to our clients throughout the course of development will reference these requirements. Software requirements are requirements derived from taking a closer look into what kind of software tools will be needed to complete our project. Since clients are unlikely to care about the low level specifics, these requirements will often be omitted during discussions with the clients. Within this section, software requirements will only be defined the first time they appear.

Some terminology used within this section may be unknown to readers. View the Glossary in section 8.0 for definitions of technical terms.

4.1 Functional Requirements

Client level requirements represent the specific items our clients want from our finished product. These requirements along with the environmental requirements outlined in section 4.3 give us the grounds for developing the functional requirements.

4.1.1 The data display module shall be a desktop application.

CR01 specifies that the data display module must function as a desktop application. A desktop application refers to software designed to run on desktop or laptop computers, typically with a graphical user interface (GUI) for user interaction. To satisfy this requirement, we have developed the following system and software-level requirements:

- **R01 -** The data display module shall be an .exe file.
- **R02 -** The data display module shall run a GUI.
    - **SR18** - The function getInput() shall be capable of listening for user input data and deciding what to do with it.
    - **SR19** - The Function clickEvents() shall be capable of redirecting the user to the events page.
    - **SR20** - The Function clickStatus() shall be capable of redirecting the user to the status page.
    - **SR21** - The Function clickElectrical() shall be capable of redirecting the user to the electrical page.
    - **SR22** - The Function clickSettings() shall be capable of redirecting the user to the settings/help page.
    - **SR23** - The Function dropboxFilter() shall be a dropdown box allowing the user to select the correct filter.
- **Use Case**
    - The User will be able to access the application through an installed application located on their computer.

4.1.2 The data display module shall read input data via RS422 serial protocol from the controller simulator.

CR02 establishes a critical communication protocol between the data display module and the controller simulator. RS422 is a standard serial communication protocol which indicates a point-to-point communication link between the display module and simulator, which emulates the behavior of the actual controller. To satisfy this requirement, we have developed the following system and software-level requirements:

- **R03 -** The *data display module* shall be capable of *deserializing* messages received via an RS422 serial port.
    - **SR02** - The functions deserializeStatus() and deserializeEvent() shall be capable of generating the original status data and event data, given a serialized bit string.

    After we have received a serialized message, we must also be able to restore it to its original format so that it may be used within the program.

- **R04 -** The *controller simulator* shall be capable of *deserializing* messages received via an RS422 serial port.
    - **SR02**
- **R05 -** The *data display module* shall be capable of *serializing* messages to send through an RS422 serial port.
    - **SR01** - The functions serializeStatus() and serializeEvent() shall be capable of generating bit string versions of given status data and event data respectably. When working with serial communication it is important to realize that computers use bit strings (strings of 1s and 0s) to store data. To be able to send any kind of data through a serial port, we must develop a method of translating data like integers and strings into bit strings
- **R06 -** The *controller simulator* shall be capable of *serializing* messages to send through an RS422 serial port.
    - **SR01**
- **Use Cases**
    - The data display module will be able to send and receive serialized messages to and from the simulated controller. The data display module will be able to deserialize these messages.
    - The simulated controller will be able to send and receive serialized messages to and from the data display module. The simulated controller will be able to deserialize these messages.

4.1.3 The data display module shall have the ability to write event data into a log file.

CR03 enables the logging of significant events, offering a historical perspective on system behavior. This log file must include information about all events that have been captured since a connection has been established. To satisfy this requirement, we have developed the following system and software-level requirements:

- **R07 -** The data display module shall be capable of generating a log file including all events up to the point in which the user requests the log file (this is the case for generating a log file *during* a session)

  - **SR08** - The function initializeEventLL() shall be capable of creating a head node for a linked list given an event string.
  - **SR10** - The function addEvent() shall be capable of adding a new event to the event linked list.
  - **SR11** - The function outputEventLog() shall be capable of writing a log file, given a linked list of event data.
  - **SR23**
- **R08 -** The data display module shall be capable of automatically generating a log file including all events that occurred during a complete session. (this is the case for automatically generating a log file *after* a session has ended)
  - **SR08, SR10, SR11**
- **R09 -** Up to 5 auto-saved event log files shall be stored in the log file folder until overwrites occur on the oldest auto-saved file.
  - **SR11**
- **Use Cases**
  - The User will be able to receive a CSV file containing all information received from the controller simulator since the connection was established.
  - The User will be able to click a ‘Download’ button to download the log file.
  - The data display module will generate the log file automatically after a complete session.
  - The data display module will generate the log file by request from the User.
  - The data display module will generate a log file containing all events that occurred to the controller simulator data during the session.
  - The data display module will automatically save five event log files.

4.1.4 The data display module shall display all weapon status information directly to the application’s window for the duration of a session.

CR04 states that the data display module must exhibit the capability to present all weapon status information directly within the application’s window throughout a session. This emphasizes a user-centric approach, prioritizing near "real"-time visibility and accessibility of critical weapon system information. To satisfy this requirement, we have developed the following system and software-level requirements:

- **R10 -** The controller simulator shall send status updates through serial port every .25 seconds.
    - **SR04** - The function sendMessage() shall be able to transmit a serialized string through the serial port. This function will be called after a serialized string has been developed, it will ensure that the message is transmitted to the other program and ensure that the other program understands the message.
    - **SR05** - The function readMessage() shall be able to detect incoming serialized messages from the serial port and update the status class with new data. ReadMessage is the concurrent process responsible for listening on the serial port and recording data received. This function will work with sendMessage to facilitate proper transportation of bitstring from one device to another.
    - **SR09** - The function initializeElectricalLL() shall be capable of creating a head node for a linked list given a set of electrical data. Then adding an unknown number of new nodes for subsequent electrical components.
- **Use Cases**
    - The controller simulator will send status updates through RS422 every .25 seconds to the graphical user interface of the application.

4.1.5 The controller simulator shall send event updates to the data display module.

CR05 outlines that all weapon status information and event updates from CR04 must be acquired via the controller simulator. To satisfy this requirement, we have developed the following system and software-level requirements:

- **R11 -** The controller simulator shall send event updates through the serial port at most 0.1 second after they are generated.
    - **SR04, SR05**
- **Use Cases**
    - The controller simulator will send the event updates to the data display module

every time the events are generated

4.1.6 The data display module shall not require admin rights to install, set up, or use.

CR06 specifies that our software product must not necessitate administrator rights for installation, setup, or usage. This establishes a sense of accessibility and convenience, allowing our product to be easily deployed and utilized without elevated permissions. To satisfy this requirement, we have developed the following system-level requirement:

- **R12 -** The data display module shall not require admin rights to install setup or use

4.1.7 The data display module shall include filtering options to filter events and errors.

CR07 mandates that our events/errors tab must include a filtering option. Because there are many different types of messages that can be displayed, the user should be able to differentiate between errors, cleared errors, active errors, and non-errors. To satisfy this requirement, we have developed the following system-level requirements:

- **R13 -** The data display module shall have the capability to display *only errors* to the Events tab of the GUI
- **R14 -** The data display module shall have the capability to display *only cleared errors* to the events tab of the GUI.
- **R15 -** The data display module shall have the capability to display *only active errors* to the events tab of the GUI.
- **R16 -** The data display module shall have the capability to display only *non-error events* to the events tab of the GUI.

Which all implement the following software-level requirement:

- **SR13** - The function filterEvents() shall be capable of filtering event data displayed in the GUI according to user preference.

- **Use Cases**
    - The User will display errors, cleared errors, active errors and non error events from the data display module and have the results displayed in the GUI.

These requirements simply exist as a precise mode of defining what our conditions of success are in this project. Additionally these requirements allow us to foster a concrete understanding with our clients as to what Team Controller's obligations are so that we may better provide the services they have requested.

4.1.8 Additional Requirements

Some software-level requirements exist that do not necessarily fall into a client-level requirement, but are necessary to ensure our product functions as expected:

- **SR03** - The connect() function shall be capable of performing handshake protocols with another device running connect(). Handshake protocols, defined in the glossary section, are essential for making sure communicating programs are on the same page. They are used to synchronize communication to prevent themselves from interrupting one another or misinterpreting data.
- **SR06** - The function disableComs() shall be able to pause serial communication.
- **SR07** - The function enableComs() shall be able to resume serial communication. The weapon controller is an important piece of equipment. It is responsible for running the weapon system, and sometimes it will be overwhelmed with this task. In this case it is ideal that communication between the controller and data display module should be stopped until the controller is less busy. These functions give the controller a way to tell the data display module that it is busy and unable to send data until further notice, and re-enable communication when it is ready to do so. Adding this functionality to our controller simulator and data display module is important so that our data display module is better prepared to work with real weapon controllers when the time comes.
- **SR12** - The function outputElectricalLog() shall be capable of writing a log file, given a linked list of electrical component data.
- **SR14** - The function randomizeEvents() shall be capable of generating random event strings out of a given set of events at random time intervals within a given range.

- **SR15** - The function randomizeStatus() shall be capable of generating random status updates every 0.1 seconds.
- **SR16** - The function customEvent() shall be capable of taking in user input data and generating an event string using this data.
- **SR17** - The function customStatus() shall be capable of taking in user input data and updating the status data structure using this data.
- **SR24** - The Function clickDownload() shall allow the user to download all the program's content into a log file.

4.1.9 Traceability Matrix

The purpose of this table is to define which system requirements satisfy each client requirement and which software requirements/functions will be used by each system requirement.

Client requirements	**System requirements**	**Software requirements**
CR01 The data display module shall be a desktop application.	**R01** The data display module shall be an .exe file.	NA
	R02 The data display module shall run a GUI.	SR18 SR19 SR20 SR21 SR22 SR23
CR02 The data display module shall read input data via RS422 serial protocol from the controller simulator.	**R03** The **data display module** shall be capable of **deserializing** messages received via an RS422 serial port.	SR02
	R04 The **controller simulator** shall be capable of **deserializing** messages received via an RS422 serial port.	SR02

	R05 The **data display module** shall be capable of **serializing** messages to send through an RS422 serial port.	SR01
	R06 The **controller simulator** shall be capable of **serializing** messages to send through an RS422 serial port.	SR01
CR03 The data display module shall have the ability to write event data into a log file	**R07** The data display module shall be capable of generating a log file including all events up to the point in which the user requests the log file (this is the case for **generating a log file during a session**)	SR08 SR10 SR11 SR23
	R08 The data display module shall be capable of automatically generating a log file including all events that occurred during a complete session. (this is the case for **automatically generating a log file after a session has ended**)	SR08 SR10 SR11
	R09 Up to 5 auto-saved event log files shall be stored in the log file folder until overwrites occur on the oldest auto-saved file.	SR11
CR04 The data display module shall display all weapon status information directly to the	**R10** The controller simulator shall send status updates through serial port every .25 seconds.	SR04 SR05 SR09

application’s window for the duration of a session.		
CR05 The controller simulator shall send event updates to the data display module.	**R11** The controller simulator shall send event updates through the serial port at most 0.1 second after they are generated.	SR04 SR05
CR06 The data display module shall not require admin rights to install, set up, or use.	**R12** The data display module shall not require admin rights to install setup or use	NA
CR07 The data display module shall include filtering options to filter events and errors	**R13** The data display module shall have the capability to display **only errors** to the Events tab of the GUI	SR13
	R14 The data display module shall have the capability to display **only cleared errors** to the events tab of the GUI.	SR13
	R15 The data display module shall have the capability to display **only active errors** to the events tab of the GUI.	SR13
	R16 The data display module shall have the capability to display only **non-error events** to the events tab of the GUI.	SR13
CR08 The system and its environment shall be installed via an installer file.	**R17** The system and its environment shall be installed via an installer file	NA

CR09 The system shall be portable on Windows 10 or 11	**R18** The system shall be portable on Windows 10 or 11	NA
CR10G The system should be portable on Debian linux distributions	**R19G** The system should be portable on Debian linux distributions	NA

4.2 Performance Requirements

Performance requirements represent specific quotas which relate to the functionality discussed in previously defined requirements. These are things like the refresh rate of our GUI, and serial communication speed.

- **PR01 -** The application shall be designed with object oriented programming principles to ensure *readability* of the application.
- **PR02 -** The application shall be designed with object oriented programming principles to ensure *scalability* of the application.
- **PR03 -** The application shall be designed with object oriented programming principles to ensure *modularity* of the application.

PR01:

Once development of the application is complete, the team will hand over the source code to developers at Northrop Grumman. To ease their development process, the application should be readable. This will include: class names, variable names, and method names. The team will include comments as needed.

PR02:

Like **PR03**, the team intends on designing the application in such a way that ensures for easy scalability. Using object oriented programming, the team will design the application to be easily scalable.

PR03:

Again, this non-functional requirement is aiming to create a modular application to assist in the hand-off to Northrop Grumman software engineers. Using OOP functionality, the application will be designed in a modular way to increase readability and functionality.

4.3 Environmental Requirements

Environment requirements represent constraints imposed by the environment in which the clients want our finished product to be deployed in. For this project our environmental requirements are as follows:

4.3.1 The system and its environment shall be installed via an installer file.

Many of the features of our project require assets such as images, supporting files and custom directories. We will need to install these assets into the host system in order for the project to operate as intended. That is where CR08 becomes a necessity. Using an installer will allow us to generate a proper environment for our system to be deployed in. We plan to create an assets directory and a log directory. Additionally a main project directory will be generated to store all installed data including our data display module and the previously mentioned directories. While running, our data display module can navigate these directories and the files they hold to perform its operations. If the user would like to uninstall the application all directories and data must be deleted using the same installer file that was used to deploy the project. To satisfy this requirement, we have developed the following system-level requirement:

- **R17 -** The system and its environment shall be installed via an installer file

4.3.2 The system shall be portable on Windows 10 or 11.

CR09 states that the client intends to run our product on computers using mainly Windows 10 and 11. We must ensure that all of the technology we use for development is compatible with these platforms. To satisfy this requirement, we have developed the following system-level requirement:

- **R18 -** The system shall be portable on Windows 10 or 11

4.3.3 The system *should* be portable on Debian linux distributions.

Our clients also informed us that some of their computer systems operate in linux environments. As a stretch goal we have decided to consider technologies that are cross compatible with windows and linux to give our clients a greater degree of flexibility when working with our product. Considering CR10G is a stretch goal, it is not required to be satisfied by our minimum viable product:

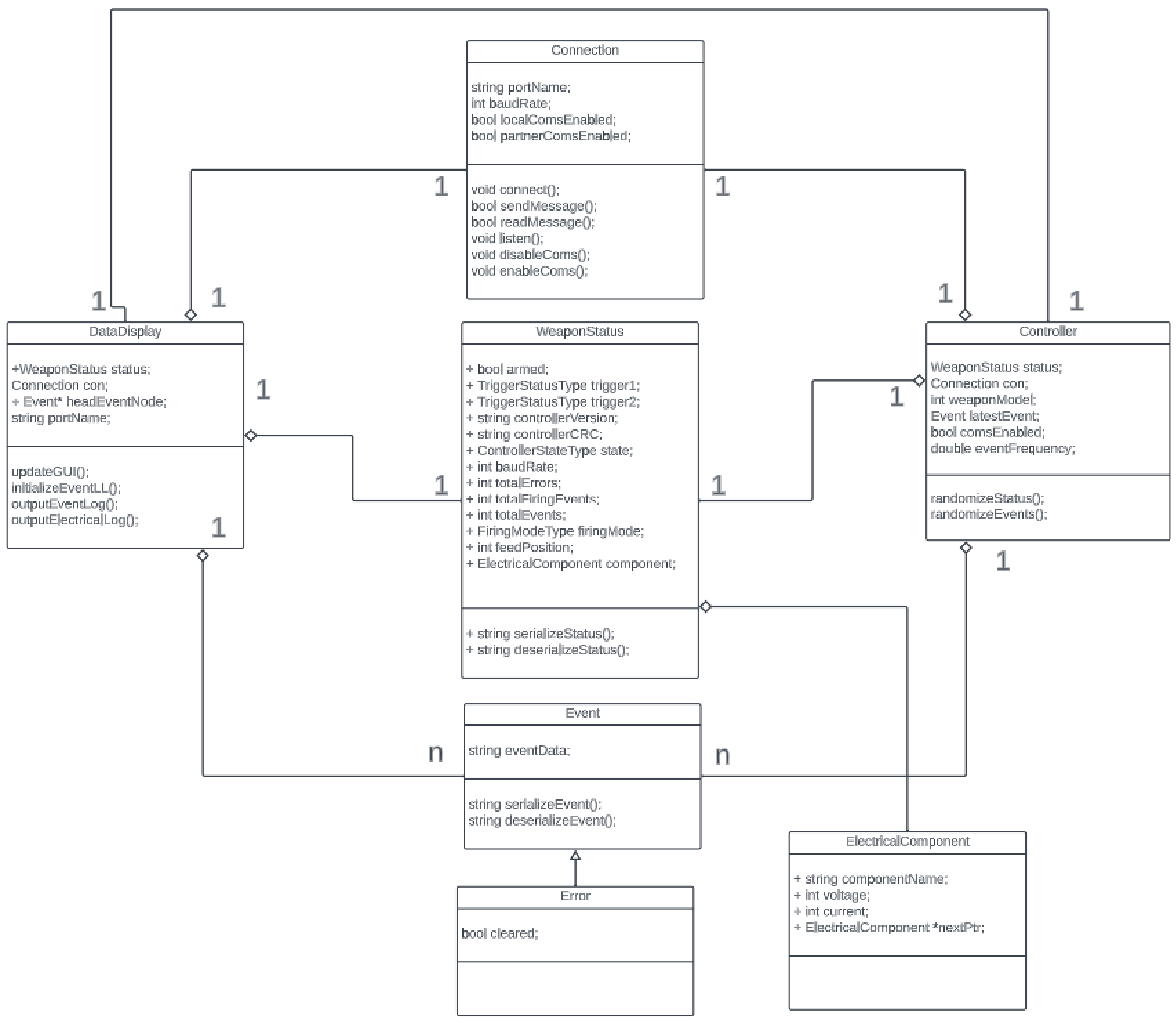
- **R19G -** The system should be portable on Debian linux distributions

4.3.4 Controller Simulator

While our clients currently have software capable of feeding weapon data to our data display module, we do not have access to this software for purposes of confidentiality. Therefore, in order to test our data display module, we must develop a controller simulator which will emulate the controller software currently in use by our client. After our project has been completed, our controller simulator along with our data display module will be handed over to Northrop Grumman for further development of the system. As it stands, the controller simulator we develop will only need to include capabilities for generating status and event data then delivering it to our data display module through the use of RS422 serial communication.

4.3.5 System Architecture

Our product will be divided into specific subsystems that all contribute to the overall project, which have been mentioned in the sections above. These subsystems are closely related to one another and there is often overlap between the functions needed for each. To give you an idea of how these systems work with one another, a class diagram will be provided along with an explanation included within each section.



Serial Communication

In order for our project to be successful, we must develop a few simple serial utility functions that will support the communication between the controller simulator and the data display module.

Data Display Module

The data display module is the brains behind the GUI. It is responsible for updating data fed into the GUI and communicating with the controller simulator to get the updated data. It must also process user inputs into the GUI and perform file output operations. Additionally it must be error resistant and possess multithreading capabilities to do all of these things at once. Considering the data display module relies heavily on the use of the serial communication functions, GUI functions and object oriented programming, the list of functional requirements exclusive to only the data display module will be quite limited in size.

Controller Simulator

As mentioned previously the controller simulator is responsible for creating and sending the data that will be used by our data display module which will be running the GUI. For complete testing capabilities, the controller simulator will need to be able to take in requests for unique data to be generated. For mass testing capabilities, the controller simulator will need to be able to generate random data continuously using random rates of generation.

Graphical User Interface

The GUI is responsible for displaying the information and allowing for user interaction. In the GUI there will be multiple buttons for the user to click on. Most of the interactions with the GUI will be buttons with a few dropdown boxes.

5.0 Potential Risks

In this section we will go over all the potential risks that might occur in our program. Overall there isn’t too much major risk and most of the risks below are very minor and simple mistakes that can occur. Each risk will be rated 1-10 and 10 being a very high risk that can occur.

5.1 Incorrect Information

The biggest risk in the project is the program displaying incorrect information. This can happen in three ways: the controller information is incorrect, the software's backend processes the data incorrectly, and the serial protocol has an issue.

5.1.1 Controller Information Is Incorrect

One way the program can display incorrect information is if the controller that the data is retrieved from is incorrect. This risk is rated at a 1 and it would be out of the teams control due to the fact that it is the hardware's issue.

5.1.2 Software Miscalculates Data

Another way the program can display incorrect data is if the software's backend miscalculates the data. This risk is rated at a 2 because it would be a very noticeable issue seen in the code and can easily be fixed. As long as the team follows all coding standards and programs everything correctly this risk shouldn't be an issue.

5.1.3 Serial Encoding/Decoding Error

Finally a way the program can display incorrect information is the serial protocol encoding and decoding has an error. This risk is rated at a 4 because it would cause lots of issues in the program if the serial protocol encodes or decodes the messages incorrectly. The serial protocol must be encoded in the correct format in the controller before being sent over to the software. Then the software must decode the message and display the data sent over. If there are any issues the software should display a message saying serial protocol error. Once again this risk shouldn't be a big issue as it takes careful programming with serial communication.

5.2 Improper port hardware

The risk is very simple and it can occur if the user is using the wrong ports/cables to use the software product. The software product will only take rs422 serial protocol and any other serial protocol will not work. This risk is rated at a 1 as it's a simple mistake that can be fixed by just using the correct cables and ports. Our product will display a message if the wrong serial communication is being used.

6.0 Project Plan

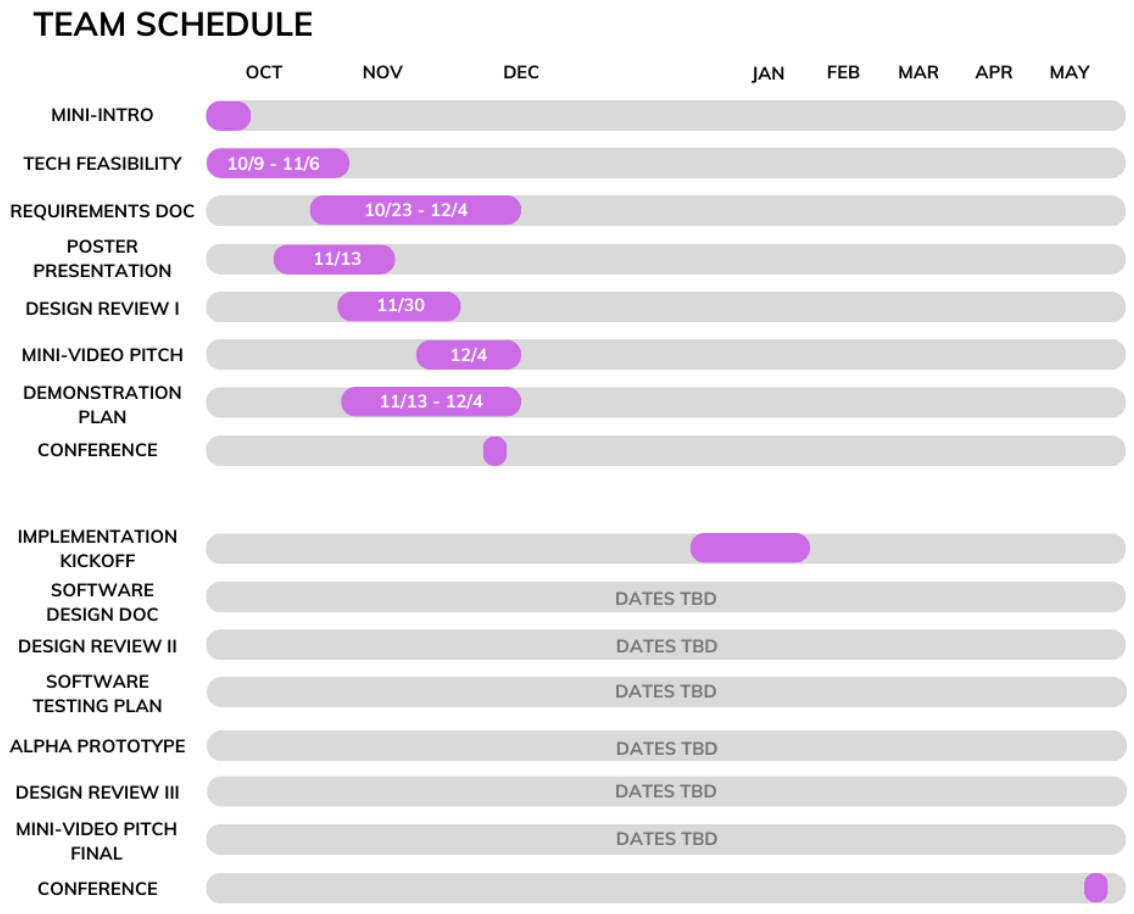
The team's project plan is segmented into 4 goals. These goals build off each other and are intended to be completed in the order detailed below.

The first milestone the team plans to achieve is the implementation of the controller simulator. This is the first milestone for a reason as it will serve as the backbone of this project. We need to simulate the data the controller will send first because it will allow the team to build the rest of the application's functionality.

The second milestone to achieve is the serial communication of the controller simulator to the data display module. This is how the real controller will connect to the diagnostic application so it is of utmost importance that the team transmit the controller simulator's data using serial communication, specifically the port we intend to use is RS422. Along with sending serialized messages, the team must also deserialize the messages coming from the controller simulator.

The third milestone is the development of a graphical user interface that can update the data that the controller simulator is sending. The GUI must update its data quickly to give the look of real time data coming in from the controller simulator. The GUI will have four screens, as shown previously in this document, so to complete this milestone, the team will have to implement all four.

The fourth milestone is the metaphorical ribbon on this project; the installer. We will need to create an installer that works without admin rights as the working environment is likely to be a tightly controlled governmental organization with strict IT usage and rules.



7.0 Conclusion

The development of our tool for Northrop Grumman's armament systems represents a crucial step towards efficiency, cost-effectiveness, and furthering technological advancement in the defense industry. The project shall respond to the current need of our clients—the demand for a sophisticated yet user-friendly solution to swiftly (and remotely) diagnose and resolve issues with their products. The current methodology of dispatching engineers to customer locations poses challenges in an era where speed, precision, and cost-effectiveness are significant factors to consider. Our proposed solution is a simple, secure, and intuitive desktop application/GUI that interfaces with weapon systems via RS422 serial communication. One of the main objectives is to empower the end-user to more effectively diagnose and potentially resolve issues independently. Additionally, the application will facilitate a remote communication between the weapon system and Northrop Grumman's engineers. With the ability to download and send diagnostic information, the process will be much more streamlined and significantly reduce response times.

This document has been meticulously created to provide a comprehensive overview of the project's scope, objectives, and technical requirements. By delving into the specifics, the team has identified several verifiable customer-level requirements, ranging from "real" time data display to the seamless integration of filtering options. These were then broken down into system and software-level requirements, which will significantly aid the team in the implementation stage of the project. As we move into these next phases, we now have a detailed blueprint to our disposal. The team anticipates positive outcomes that actively contribute to the maintenance and optimization of Northrop Grumman's weapon systems worldwide.

8.0 Glossary

8.1 Project Related Terminology

System - All files developed for the purposes of satisfying the client level requirements.

- **Data display module** - displays status updates and events to the user
- **Controller simulator** - Generates status updates and events then sends via RS422 serial protocol to the data display module.
- **Event Log file** - Will be generated by the data display module to contain 1 or more event strings encountered during a session.
- **Installer** - will deploy the project in the customers system and perform environment setup and initialization. Will record initial user preferences and take them into account during installation. I.e. "Do you want a shortcut on your desktop?"

Environment - The directory our system will be placed in and all of its contents.

Status information - (See class diagram for specifics) General data pertaining to the weapon which can be measured at any point in time during session.

Status Update - serialized version of status information to be sent to the data display module from the controller simulator.

Event String - A string of the format "<time> <event message> <param 1> <param 2> <param 3>" generated by the controller simulator to simulate the occurrence of a weapon related event. The parameters can be NULL, but the event message must be specific text outlining what the event is. Ex. "[00:12:41] Weapon overheat 237 200" where 237 represents measured barrel temp and 200 represents max recommended barrel temp in degrees celsius.

Event update - The serialized version of the event string to be sent to the data display module from the controller simulator.

Session - The time measured from the moment the controller sim is connected to the data display module to the moment the controller sim is disconnected from the data display module.

Electrical component data - A data structure containing float values for current and voltage and a name for the component.

8.2 General Definitions

RS422 - A specific type of serial communication technology including specialized serial ports for wired connections between devices.

Serial Port - A hardware connection port built to facilitate serial communication with outside devices.

Linked list - A type of data structure which stores a node containing any kind of data along with a memory address which holds the location of the next node in the list.

Handshake Protocols - A necessary set of agreements between two devices before they perform serial communication.

9.0 References

[1] Northrop Grumman. (2023). *B-2 Spirit Stealth bomber*. What We Do - Air. https://www.northropgrumman.com/what-we-do/air/b-2-stealth-bomber

[2] Northrop Grumman. (2023). *James Webb Space Telescope*. What We Do - Space. https://www.northropgrumman.com/space/james-webb-space-telescope

[3] Northrop Grumman. (2023). *Global Presence*. Who We Are. https://www.northropgrumman.com/who-we-are/global-presence